# 分布式事务

## 分布式事务产生的原因

1. 数据库分库分表。由于库不是同一个（不同的库拥有自己本地事务），不能使用本地事务来保证。
2. SOA(业务的服务化)。跨服务调用。

总结：1、跨JVM调多个数据库实例（远程调用，且不是同一个本地事务）；2单系统调用多个数据库实例（不是同一个本地事务）；3多服务调用同一个数据库（远程调用）

## CAP

### 2.1 CAP介绍

**C**（Consistency）:一致性:多个节点读取数据时都是最新状态（主从一致），即用户在任何时刻查询主从数据库数据都要一致（操作**同步**完成后才能允许查到最新结果）。

--同步加锁，操作结束才会释放锁，存在主从延迟，数据强一致性。

**A**（Availability）:可用性:任何事物操作都能立即响应数据，允许返回历史数据，但是不出现用户操作失败或者访问超时等用户体验不好的情况。。（会出现一定时间内的数据不一致，明显不能满足C一致性要求）

--不能保证数据一致性的C锁定，不能返回错误日志。允许返回新或者旧数据。

**P**（Partition tolerance）:分区容错性: 分布式系统在遇到某节点或网络分区故障的时候，仍然能够对外提供满足一致性或可用性的服务。

--异步操作，一主多从，集群。是分布式系统的基本能力。

### 2.2 CAP组合（三进二）

CAP三个特性只能满足其中两个，那么取舍的策略就共有三种：

AP:分布式系统（如订单、到账等，一定时间内一致即可）一般是AP。放弃强一致性。

CP:银行系统（整个事务要一起完成）一般是CP。放弃可用性。

CA:数据库一般选择的是CA，放弃分区容错性。严格来说不是标准的分布式系统。

## BASE理论

强一致性：复制是同步的，需要完全同步后才能查询。（像银行操作时有一段时间等待后才能操作）。

最终一致性：复制是异步的，主从异步复制，允许查到旧数据。一段时间主从库复制完成。查询数据为最新数据。

BASE:基本可用、软状态、最终一致性。

基本可用：分布式系统的问题，可以部分服务丢失，但是核心功能要保证可用。（如系统压力巨大时，可以允许不能下单，但是产品要可查）；

软状态：中间状态（如秒杀时等待状态，完成后更新为实际状态成功或者失败）；

最终一致性：一段时间主从库复制完成。查询数据为最新数据（如等待状态最终变为成功或者失败）。

## 分布式事务方案之2PC

2PC:两阶段提交协议。2是两阶段，即P的准备阶段，C的提交阶段。

准备阶段：没有提交事务，记录修改前数据与修改后数据。

提交阶段：回滚或者提交事务后释放锁。

核心：加了一个事务管理器来协调多个本地事务。

### 2PC之XA方案

AP:应用

RM:资源管理器，事务的参与者（本地分支事务）。

TM:事务管理器（全局事务），负责提交与回滚事务。

DTP：处理模型(跨库类型，统一事务标准)，DTP模型定义TM和RM之间接口的规范叫XA。基于2PC的接口协议与基于数据库的XA协议来实现2PC的XA方案。

特点：

1. 需要数据库支持XA事务；
2. 资源锁要两阶段结束后才能释放（同步），性能较差。

### 2PC之Seata方案

Seata是阿里开源的Facar，后改名为seata。是开源的分布式事务框架，不需要数据库支持XA。是基于应用层控制的中间件，不会长时间占用连接资源、OA接入。

AT与TCC

## 备注

**Consistency** [kənˈsɪstənsi] n.一致性;

**Availability** [əˌveɪlə'bɪləti] n.可用性

**Partition tolerance** [pɑːrˈtɪʃn] [ˈtɑːlərəns] n. 分区容错性